Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

EVALUATING CONTAINER STACKING RULES USING SIMULATION

Eelco van Asperen

Center for Maritime Economics and Logistics
Erasmus University Rotterdam
3062 PA Rotterdam, THE NETHERLANDS

Bram Borgman
Rommert Dekker

Econometric Institute
Erasmus University Rotterdam
3062 PA Rotterdam, THE NETHERLANDS

ABSTRACT

Container stacking rules are an important factor in container terminal efficiency. In this paper, we describe a discrete-event simulation model that has been used to evaluate online container stacking rules. We build on prior research and demonstrate that results obtained for smaller stacking areas are also valid for a larger stacking area. The use of information regarding container departure times (even if is imperfect) is shown to be more beneficial than the use of exchange categories. Stacking rules that take the workload of the automated stacking cranes into account outperform rules that do not. The experiments conducted with the simulation model show that it can capture the amount of detail required and that it is flexible enough to support the evaluation of the stacking rules.

1 INTRODUCTION

Marine container terminals play a central role in the worldwide distribution of goods that is essential to global supply chains. These container terminals link sea transport via container ships to land transport via trains, barges, and trucks. One of the main problems in container terminals concerns the stacking of containers. Although it is also one of the main advantages of containers, viz. that they can be stacked on top of each other, additional work is required if the bottom container is needed. In that case the top containers have to be moved to another place, which is called a reshuffle or unproductive move.

Accordingly, every terminal needs a stacking strategy. The main objectives of such a strategy are 1) the efficient use of storage space, 2) limiting transportation time from quay to stack and beyond, (and vice versa), and 3) the avoidance of reshuffles. Of course, the importance of each criterion depends from terminal to terminal. Ports like Singapore and Hong Kong have limited land space, so they need efficiently used storage spaces. Note also that these objectives are conflicting: you cannot maximize them all. For example, the third objective would be optimized by having stacks of only one container high; however, this would lead to very inefficient use of storage space.

This paper focuses primarily on the short term decision to allocate an incoming container to a stacking position. We consider an import container terminal with a high uncertainty regarding the departure time of the containers, such as terminals in Europe (Hamburg–Le Havre range) and the USA. We try to mimic the most common situation where imperfect or imprecise information about the departure time of a container is available. Moreover, we consider online stacking rules, which do not require extensive computations and can be used in many types of stacks and for large numbers of containers. The type of container terminal we consider uses a high degree of automation. This typically implies a tight coupling of the various activities and a reduced flexibility to handle unexpected circumstances. Also, the terminal under consideration has both a high yard utilization and a high berth utilization. This reduces the potential to optimize the storage yard by performing housekeeping moves at times of low utilization. Careful evaluation of the terminal design, the operational procedures, and especially stacking rules is therefore required.

We concentrate on the trade-off between traveling and finding a position which limits the likelihood of reshuffles. We use a quite realistic simulation program to test our ideas. As benchmark we take both random stacking policies as well as policies which use precise information on the container departure times.

The paper is structured as follows. We start with giving an overview of existing literature on stacking in section 2. The basic stacking concepts are explained in section 3. Next we explain the set-up of our simulation model in section 4. The experimental set-up is presented in section 5, while section 6 presents our benchmark algorithms. The results from the experiments are given in section 7 and we finish the paper with conclusions in section 8.

2 LITERATURE REVIEW

Academic literature on stacking problems is not very common yet, perhaps because the problem does not easily lend itself to analytical solutions (Dekker, Voogd, and van Asperen 2006). However, in recent years, the subject seems to get more attention, because its importance is recognized (Steenken, Voss, and Stahlbock 2004). In a recent overview paper on operations research at container terminals, Stahlbock and Voss (2008) looked at a number of aspects of container terminal operations. Among the topics surveyed were stowage planning, berth allocation, crane optimization, terminal transport optimization, and storage and stacking logistics. Their work is an extension of an earlier overview (Steenken, Voss, and Stahlbock 2004), which also contains a paragraph on how stacking is done in practice.

Various methods are used to tackle the stacking problem, but two main approaches can be distinguished like in job scheduling (Dekker, Voogd, and van Asperen 2006). *Analytical calculations* with full information on the moment a container will be retrieved from the stack. They are often based on integer programming and take relatively much computation time. Next there are *detailed simulation studies* which evaluate various stacking strategies. These strategies can be *online*, in which they determine for each container separately where to place it independent of other incoming containers and *offline*, where locations are found simultaneously for all containers to be offloaded from a ship. So far only online rule-based strategies have been studied in simulation studies. These rules can handle imperfect or imprecise information on departure times of containers. Their study takes a lot of time and the results may be dependent on the simulation set-up. Dekker, Voogd, and van Asperen (2006) distinguish two types of stacking strategies: category stacking and residence time stacking. The former strategy assumes that containers of the same category (e.g. having the same size, destination, weight, etc.) are interchangeable, and can thus be stacked on top of each other without the risk a lower container in a stack is needed before the ones on top of it have been removed. The latter strategy does not use categories, but instead looks at the departure times of the containers: a container can only be stacked on top of containers that all have a (planned) departure time that is later than the departure time of the new container.

Detailed simulations were performed by several authors. Dekker, Voogd, and van Asperen (2006) simulated different stacking policies for containers in an automated terminal. In particular, several variants of category stacking (with up to 90 different categories) were examined and compared with a base case in which containers are stacked randomly. The simulations demonstrated very high peak workloads during the handling of very large container ships. Category stacking was found to significantly outperform random stacking. Considering the workload of each automated stacking crane (when selecting the lane for an incoming container) and stacking close to the export transfer point were found to provide additional performance benefits. There was no significant benefit to using category stacking for containers with onward transport by short-sea/feeder, rail, truck, or barge. As the category definitions are based on information used in stowage planning, they advocate an integration of terminal operations and stowage planning. For a more detailed analysis, Borgman, van Asperen, and Dekker (2010) use a similar model and similar throughput data with a smaller stacking area. They evaluated the trade-off between the movement of the automated stacking crane and the possibility of more reshuffles. They found that even imperfect or imprecise information regarding the departure times is valuable and that minimizing the difference between departure times is beneficial.

Duinkerken, Evers, and Ottjes (2001) also used simulation and category stacking, albeit with only a limited number of categories. Several (reactive) reshuffling rules were tested. Reactive reshuffling means the reshuffling is done when a container on which other containers are stacked is demanded for retrieval, leading to a number of reshuffling operations. This in contrast to “proactive” reshuffling, which is done when stacking cranes are idle. They evaluated several reshuffling rules (random, leveling, closest position, and minimizing remaining stack capacity reduction). The use of categories was compared with a model that required specific containers and found that the categories led to much better performance. Also, it was shown that the remaining stack capacity strategy lead to big improvements when compared to the other three. Duinkerken, Evers, and Ottjes (2001) also tested two “normal” stacking strategies (i.e. for when a container has just arrived), namely random and with dedicated lanes for a quay crane. However, using dedicated lanes is hard to do in practice, as load plans are not known in advance. Also, this strategy did not yield much improvement over random stacking.

Park et al. (2006) used simulation to determine the best combination of an automated guided vehicle (AGV) dispatching rule with a reactive reshuffling rule, for various amounts of containers and AGV's in an automated container terminal. Their goal is to minimize the number of reshuffling operations. Park et al. used the random, closest position (but with residence time taken into account) and minimal RSC reduction reshuffling rules from Duinkerken, Evers, and Ottjes (2001). In most cases, the minimal RSC reduction rule, combined with the Container Crane Balancing (CCB) dispatching rule (the AGV is sent to the crane which has the most containers waiting for it), lead to the least reshuffling operations.

The design of simulation models for container terminals, and the detailed algorithms used for stacking in practice, are generally not well documented. Sgouridis and Angelides (2002) describe a detailed model of a container terminal created using FlexSim. The stacking algorithm is, however, limited to random stacking only. Petering (2010) uses a detailed model of a transshipment terminal with manually operated equipment to evaluate different methods for controlling dual-load yard trucks in real-time, but does not specify the stacking method used.

3 BASIC CONCEPTS

In this section we present some generic concepts that form the basis of the stacking rules we will evaluate in this paper. The precise formulations will depend on the layout of the stacking area. While we only present these formulations for one particular layout (in order to clarify the presentation and analysis), the formulations can be adapted for other layouts.

We want to investigate the basic concepts of stacking containers in a yard. The core dilemma is that we would like to stack a container that arrives and departs at the quay-side as close to the transfer point quay-side as possible because this will minimize the total travel time of the stacking crane when the container enters and exits the stack. As there are many of these sea-sea containers, this would require us to stack high. Unfortunately, when we start to stack containers on top of each other, we face the risk of stacking a container on top of a container that will depart before the incoming container. This will lead to a reshuffle, which takes time. We will thus have to balance the travel and hoisting time of the stacking crane and the time taken by reshuffle moves.

If we consider a single lane of the type of container terminal under investigation, we see that there is a single rail-mounted stacking crane that has to perform all the stacking moves for that lane. In this front-end interchange design there is an interchange point between the horizontal transport (AGV or straddle carrier) and vertical transport (by the automated stacking crane or ASC) at either side of the lane. We can distinguish between containers that are moving into the lane (i.e., that are being stacked) and containers that are moving out of the lane (they are being "unstacked"). Containers can enter and leave the lane at two sides: at the quay side (for containers that are coming from or going to deep-sea ships) and at the land side (for all other modes of transport). Figure 1 provides a schematic overview of the terminal layout. The layout of this terminal has the stacking lanes perpendicular to the quay. Each lane has a length, a width, and a height. We will refer to a single line along the length of the lane as a lane segment.

A trade-off that is worthy of investigation is the trade-off between the time it takes the ASC to travel to a certain location and the amount of time required to (un)stack a container. For a container that has arrived on a deep-sea vessel and that will also depart on another deep-sea vessel, it is attractive to stack it as close to the transfer point at the quay side as possible. If we can stack the container close to the transfer point we save travel time of the ASC both when the container is stacked and when it is unstacked. In order to have space to stack sea-sea containers close to the quay-side, we stack land-sea and land-land containers close to the land side.

Finally, we would like to evaluate the use of category stacking (as per Dekker, Voogd, and van Asperen (2006)), where a number of equivalence-categories are defined. The categories are used during the loading phase of a container vessel to pick the next container to be taken from the stack. All containers in the same category are considered to be equivalent and thus the stacking algorithm can choose among all containers of that category to pick the container with the most favorable position.

The overall approach of the experiments in this paper is focused on the operational decisions that have to be made by terminal operators. Specifically, we take the arrivals and departures that are specified as part of the generator output and perform these operations. There is no global optimization or explicit planning; the operations are performed one at a time, i.e. in a greedy fashion, whenever a container arrives. We do not consider future events such as other incoming containers.

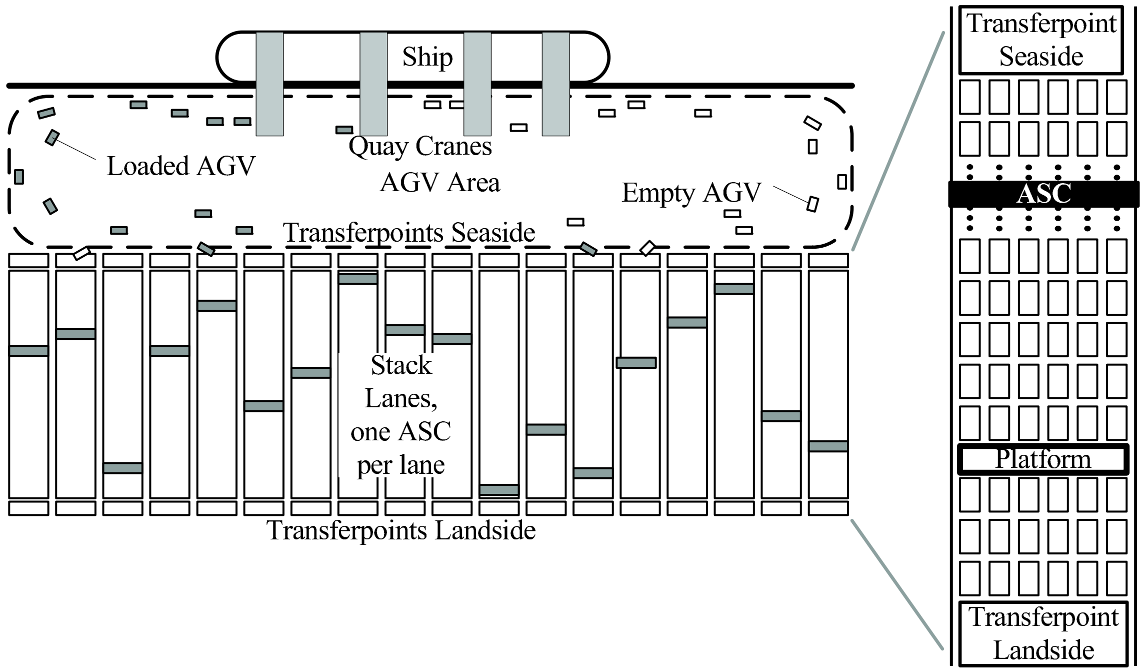

Figure 1: Terminal Layout with details of a single lane (Dekker, Voogd, and van Asperen 2006)

4 SIMULATION MODEL

The simulation model that was developed for the experiments in this paper consists of two major components: a generator and a simulator. Both programmes have been implemented in the Java programming language; the simulator uses the SSJ discrete-event simulation library (L'Ecuyer and Buist 2005).

The generator program creates arrival and departure times of the containers. The generator is based on the same data as the generator in (Dekker, Voogd, and van Asperen 2006), including sailing schedules and a modal-split matrix. The output of the generator is a file that contains the ship arrivals, details of the containers to be discharged and loaded, and the specification of the destination of each container. The departure time is specified as the planned (a.k.a. expected) departure time and the actual (a.k.a. real) departure time. The destination can be another deep-sea vessel or (for import containers) a short-sea vessel, barge, train or truck. For each container the location of the individual container within a ship is specified. The generator takes the detailed quay crane sequences for loading and discharging into account.

The simulator program reads the output of the generator and performs the stacking algorithms. The core of the simulator itself is deterministic: the stochastic components are in the generator and, optionally, in the stacking algorithm. This setup facilitates a comparison of stacking algorithms as any changes in the statistical output of the simulator must be caused by the stacking algorithm. We will provide an outline of the simulator below. More details of the generator and simulator are described in (Borgman 2009).

4.1 Outline of Simulation Model

The event-based simulation model has been built around the following entities and resources.

4.1.1 Ground Positions, Reefer Platforms, Piles, and Stack Locks

A stacking lane is divided into several *ground positions*. Ground positions have the size of one TEU container: 20 feet long and 8 feet wide. (A TEU or Twenty-foot equivalent unit is a standardized type of sea container.) In addition to ground positions for stacking, a position can also be occupied by a reefer platform. These platforms are (in the model) also one TEU in size. The locations of reefer platforms can be set as a parameter of the simulation. (We do not use these reefer platforms in the experiments in this paper.)

A ground position may be empty or be occupied by a *pile*. A pile is a stack of at least one pile slot and at most the maximum stacking height (a configurable parameter). Piles contain a single size of containers and can use a number of ground positions. A pile with 20-feet containers uses one ground position, a pile with 40-feet containers uses two adjacent ground positions, and a pile with 45-feet containers uses three ground positions.

Piles consist of *pile slots*, which contain information on whatever is present in that slot. In particular, the pile slot contains a container (which may or may not be already stacked - see below), the type of stack lock present (if any), and the (expected) unlock time (if any).

4.1.2 Automated Stacking Cranes (ASCs)

ASCs are the most complex resources in the simulation. This is mainly because they have a big influence on the results of the simulations, which means it is important to model them in detail, more so than AGVs for example. The ASC process is quite involved and has interactions with several other processes.

The ASCs pick up containers from the Transfer Point Landside (where a straddle carrier put them previously) or from a waiting AGV at the Transfer Point Quayside. The ASC then moves these containers to their specified position in the stacking lane. In case of containers leaving the stack, the ASC picks up containers from the stack and takes them to the Transfer Point Landside or to a waiting AGV at the Transfer Point Quayside. Alternatively, the ASC may move containers between locations in the lane, a process known as remarshalling. This is particulary necessary when a container, which is not at the top of its pile, has to be moved to a Transfer Point.

For each ASC we maintain a task queue. The tasks in the queue are executed in the order of arrival. Every task is made up of several subtasks. There are two types of subtasks: move-subtasks and hoist-subtasks. Move-subtasks tell the crane to move to a certain position in the lane, while hoist-subtasks tell the crane to lift a specific container (which they have a reference to). ASCs are not interrupted in their tasks: when a task is started it has to be finished before a new one is started. It is, however, possible to change a task while it is being executed. This is done when a container, which has not been stacked yet, receives a new location.

An ASC can be either in raised or lowered position. Raising and lowering takes a fixed amount of time. An ASC can only move in raised state, and has to be lowered to set down or pick up a container.

4.1.3 AGVs and Straddle Carriers

AGVs and Straddle carriers serve the purpose of transporting containers from the transfer points at either end of the stacking lane to the quays and truck loading points (and vice versa), respectively. Straddle carriers are straightforward to model, since they just pick up and drop off containers without any involvement needed by the ASC. If a container to be picked up is not yet at the landside transfer point, the straddle carrier waits at the transfer point until it has been delivered by the ASC.

AGVs are a little more complex to model, since they require the ASC of a lane to pick up the container they are carrying, or alternatively to put a container on them. To coordinate the various tasks for the AGVs and straddle carriers, two separate classes keep a list of tasks and assign them to vehicles whenever they have finished a task, or immediately to the first waiting vehicle if there were any idle vehicles.

4.2 Time and Delays

The simulation uses hours as the base time unit. Hence, if a container is to be picked up 10 hours after its arrival in real life, it takes 10 time units in the simulation. Likewise, if a crane movement takes 36 seconds in real life, it takes 0.01 time units in the simulation. A special event is scheduled after the warm-up time to reset all counters and statistics.

In these experiments, we have not modelled perturbations such as machine breakdowns as the purpose of this model is to find the best stacking rules.

4.3 Input Data

The inputs can be divided into three groups: 1) Container arrivals and departures, 2) Stack layout, and 3) Simulation parameters.

4.3.1 Container Arrivals and Departures

We have separated the generation of container arrivals from the main simulation program. A separate program, or *generator*, makes a list (a plain text file), which is then used as an input for the simulation. The arrivals text file is stripped of any comments and stored in an internal list. This list is then used to generate the arrivals in the simulation program. Each element of the list is then processed to generate the arrival and departure events.

4.3.2 Stack Layout

In this category, everything related to transporters and layout of the stack is included. For transporters (automated guided vehicles and straddle carriers), we have to specify the number of transports, the travel times, and the time required to load and unload a container. The ASCs require more detailed specifications (horizontal and vertical speed, including the acceleration, and times required to pick up and release containers).

The stack layout is described in terms of the number of lanes, their length and width (this implementation assumes a uniform layout), the maximum stacking height, and the locations for reefer containers and platforms.

4.3.3 Simulation Parameters

For an experiment, we can select the stacking and remarshalling algorithms, the length of the warm-up and simulation run, the number of replications, and the degree of visualisation (if any). These parameters are conveniently specified in a spreadsheet.

4.4 Reporting and Data Generation

For every batch run of simulations, the simulation model generates a trace log of events, a log file, and a report with aggregated results and a summary. These are all plain text files. A separate spreadsheet is available to facilitate the comparison of results from a number of experiments.

5 EXPERIMENTAL SETUP

The experiments in this paper all use the following configuration. Experiments are run for a 15 week period, of which three weeks are used for warm-up (to initialize the stack). We have used the generator program, discussed in section 4, to generate an arrivals file containing 138,330 arriving containers during the 12 weeks of simulation (43.8% 20ft, 49.5% 40ft, and 6.7% 45ft). As some stacking rules have a stochastic component (such as selecting a position at random), we use ten replications to get statistically robust results. These replications are used to compute the 95%-confidence intervals of the means. (For brevity, we only report the means here. All differences reported are statistically significant.)

We assume that there are sufficient AGVs and straddle carriers to ensure that these resources do not act as bottlenecks. The basic configuration for the stacking area was adapted from the earlier work by the same authors (Borgman, van Asperen, and Dekker 2010), but resized to accommodate the increased number of containers and the various container sizes. The new stack has 19 lanes, each consisting of 6 segments, which in turn have 48 positions of 20 ft. The length of the lanes was increased because the average size of a container increased. Keeping the lanes short would thus result in less containers per lane, and hence less work for the ASCs. The increase in the number of lanes was still required to accommodate the increase in the absolute number of containers. Each lane still has a single ASC per lane, and the maximum stacking height is still three containers. We denote the described configuration as $19 \times 48 \times 6 \times 3$ (19 lanes, each lane being 48 TEU long, 6 segments per lane, and a maximum stacking height of 3 containers). Here, the aim is not to provide an exact model of the actual terminal on which this simulation model is based but rather to provide a test-bed for the evaluation of stacking rules.

The average residence time of a container is 3.8 days; the 90%-percentile of the dwell time is 5.3 days, and the maximum dwell time is 8 days. The average utilization of the yard is 62%.

We will use Random Stacking (RS) and an implementation of the Leveling algorithm (LEV) described in (Duinkerken, Evers, and Ottjes 2001) and (Borgman, van Asperen, and Dekker 2010) as benchmarks for the experiments.

6 BENCHMARK ALGORITHMS

In this section we introduce the basic algorithms used for comparison in the experiments.

6.1 Random Stacking

Random stacking is a straightforward way of determining a stacking position for a new container. Basically, the new container is placed at a randomly chosen allowed location, with every allowed location having an equal probability of being chosen. This algorithm is also applied for reshuffling, with the difference that we then only want to search the lane the container is in.

6.2 Leveling

The idea is to fill lanes in layers, so that all empty ground positions are filled with containers first, before containers are stacked upon others. The stacking lane is filled from the transfer point quayside on. To effectively handle different sizes of containers, we dedicate lane segments to a single size. This algorithm is also applied for reshuffling, with the difference that we only search the lane the container is in. The algorithm is:

1. Choose a random lane segment, dedicated to the required container size, with at least one available position.
2. Search for the first empty location, from the transfer point quayside towards the transfer point landside, row for row (i.e. widthwise).
3. If found: stack there
4. If not found: search all existing piles (of the same size and type), from the transfer point landside towards the transfer point quayside, row for row, for the lowest (i.e. search lowest piles first) stack location and stack on the location found first.

7 EXPERIMENTS

In this section we will present our experiments with a number of stacking rules. The performance of a stacking algorithm is measured with the following statistics:

Exit Time (ETQ and ETL). The exit time is the time (in hours) it takes to remove a container from the stack and have it ready for onward transport (to the quay or to a truck/train/barge). This time is measured for each side (quay-side and land-side) of the stack and will be listed as ETQ and ETL respectively. The exit time is the main performance indicator for a stacking algorithm. It is negatively influenced by stacking further away from the exit point and by any reshuffles that are needed to retrieve the container. When a container enters the stack, the time it takes to perform this operation is determined by the workload of the ASC (how many jobs are in the current job queue) and the time it takes the ASC to move the container to its position. There are no reshuffles when containers are stored in the stack; reshuffles only occur when a container has to leave the stack.

ASC Workload (ASC). The automated stacking cranes are critical components for the overall performance so we measure the percentage of time that the ASC's are busy. (The ASC workload will be denoted as ASC in the results.)

Reshuffles (RDC and ROC). For the unproductive reshuffle moves, we measure the number of reshuffles (denoted as RDC) as a percentage of the total number of container movements. To get an indication of the number of reshuffles that happen per move, we also measure the reshuffle occasions (as a percentage of the total number of container movements, denoted as ROC); a single reshuffle occasion implies one or more reshuffles. These numbers are not absolute indicators of performance as the time of the reshuffle is not taken into consideration. A reshuffle that occurs when the workload is low has less impact on the overall performance than a reshuffle during a peak workload, for example when (un)loading a very large vessel.

Ground Position Usage (GPU). We report only the average percentage of ground positions that are in use (denoted by GPU) as the various stacking strategies have differing preferences for stacking on the ground.

7.1 Basic Online Rules for a Large Stack

First, we test our new configuration with the same algorithms tested in (Borgman, van Asperen, and Dekker 2010). This is done to see whether the conclusions made earlier still hold when using a bigger stack with multiple types of containers. The algorithms tested and compared include, besides the base algorithms random stacking and leveling:
RS-DT: The Random-Stacking with Departure Times algorithm searches for a random pile where the top container's departure time is after the new container's departure time.
RS-DTC: This is the same algorithm as RS-DT but with a small number of classes of departure times.
LDT: The Leveling with Departure Times algorithm modifies the basic leveling algorithm to only stack a

container on top of a container with a later expected departure time.
LDT-DTC uses a small number of departure time classes rather than the actual departure time.
TPRL: The Transfer Point Random Level algorithm picks a level (height) at random and then looks for a location that is closest to the transfer point.
TVR: The Traveling Distance versus Reshuffling algorithm balances the two components.
TVR-DTC modifies the TVR algorithm to use a small number of classes rather than the actual departure time.
TVF-DTC-MD extends the TVR-DTC algorithm to minimize the difference in the departure time class.

All leveling-based algorithms (e.g. LDT and TVR) use the lane segment allocation from the leveling algorithm.

The results can be found in Table 1. (The mixed and unmixed variants of the algorithms indicate whether transshipment (or sea-sea) containers are mixed in the same piles as import or export containers. We have included the numbers used for the experiments in (Borgman, van Asperen, and Dekker 2010) for easy comparison.) From this table, we can see that the main conclusions made in (Borgman, van Asperen, and Dekker 2010) still hold. The best performing algorithm is still LDT, but of the algorithms that do not use perfect knowledge of arrival and departure times, TVR-DTC-MD scores best, again confirming the earlier results. The other results further confirm the consistency of the earlier results, regarding the relative performance of the algorithms.

It is worth noting that the differences between the various algorithms is somewhat smaller in our results (compared to (Borgman, van Asperen, and Dekker 2010)). This is probably due to the lower pressure on the stack due to the increased container size, and the resulting lower ASC workloads.

7.2 Category Stacking (CATS)

Category stacking, as described by (Dekker, Voogd, and van Asperen 2006), is successful in improving container stacking efficiency whilst using only a limited amount of extra information. Category stacking uses the principle that, on large vessels e.g. on the routes to China), the exact order of containers in the ship is not important. For trucks, on the other hand, this order is important, because every single container must reach a specific destination with a specific truck. Our implementation of this algorithm is modeled to be identical to the experiment B of (Dekker, Voogd, and van Asperen 2006), meaning only containers departing by jumbo or deep sea vessel are interchangeable. This facilitates a comparison with that earlier work and serves as a base point for further research.

The results of category stacking can be found in Table 2. The CATS algorithm performs better than the base cases, random stacking, thereby confirming the earlier results from (Dekker, Voogd, and van Asperen 2006), and leveling. Category stacking also clearly outperforms standard TVR, however, the various residence time-based algorithms (LDT-DTC and TVR-DTC-MD are shown here) all yield much better results. Therefore, we can conclude that although the information in the categories for the CATS algorithm is valuable, getting a good estimate for departure times of containers can lead to better results. We suspect this is mainly due to the fact that category stacking, as we tested it, only applies for containers exiting via jumbo or deep sea ship. In our experiments, these containers constitute about 45.5% of the total number of containers.

7.3 Crane Workload

The final set of experiments in this paper deals with the issue of lane selection. The previous experiments were all done with random lane selection. Here, we try to improve the results by instead selecting a lane with the shortest ASC work queue. This should lead to faster container throughput, because long queues are less likely to form.

The results can be found in Table 2. (Again, the numbers in the "Exp" column refer to the numbering used in (Borgman, van Asperen, and Dekker 2010). Here, we have added category 6 for Category Stacking.) The variants of the algorithms using crane workload for lane selection (marked "CW" in the table) all outperform their counterparts not using this information, even though the difference is sometimes rather small (e.g. with TVR-DTC-MD), probably because there is simply not a lot of room left for improvement. A further interesting observation is that for all experiments using crane workload, the average ASC workload turned out higher than when not using it, together with the number of reshuffles, even though the exit times are lower. This effect occurs because the extra "bad" decisions (leading to reshuffles) are offset by the better utilization of the ASCs.

Overall, it is a good idea to use the crane workload information when selecting a lane, rather than doing it randomly.

Table 1: Results of experiments with stack layout: $19 \times 48 \times 6 \times 3$. The "90%" values are the average 90% percentile values. "p" is used to indicate the values for the reshuffle penalty.

Exp.	Description	ROC	RDC	GPU	ASC	ETQ	ETL	90%	90%
		%	%	%	%	hrs.	hrs.	ETQ	ETL
0	RS	63.88	95.27	84.75	51.89	0.26	0.23	0.49	0.41
0	LEV	56.60	60.64	99.41	48.10	0.23	0.20	0.42	0.34
1	RS-DT (exp)	21.08	29.66	72.55	43.41	0.17	0.16	0.29	0.26
1	RS-DT (real)	17.05	25.24	72.72	43.00	0.16	0.16	0.27	0.25
1	LDT (mixed, exp)	7.06	7.54	73.95	39.70	0.16	0.15	0.26	0.24
1	LDT (unmixed, exp)	6.31	6.72	74.86	39.49	0.15	0.14	0.24	0.21
1	LDT (mixed, real)	0.21	0.25	74.21	39.07	0.15	0.14	0.24	0.22
1	LDT (unmixed, real)	0.34	0.42	75.15	39.05	0.14	0.13	0.22	0.20
2	RS-DTC (exp)	52.47	76.08	81.00	49.21	0.23	0.21	0.43	0.36
2	RS-DTC (real)	52.90	76.82	81.03	49.26	0.23	0.21	0.43	0.36
2	LDT-DTC (unmixed, exp)	7.17	8.07	88.12	41.39	0.16	0.15	0.25	0.22
2	LDT-DTC (unmixed, real)	6.98	7.91	88.16	41.35	0.16	0.15	0.25	0.22
3	TPRL	54.59	82.85	87.34	46.29	0.22	0.19	0.37	0.30
3	TVR (p=0)	52.54	81.11	92.54	45.41	0.20	0.17	0.34	0.27
3	TVR (p=0.04)	52.77	69.76	99.34	45.93	0.20	0.17	0.34	0.28
5	TVR-DTC (p=0.01)	11.62	15.78	89.71	40.84	0.14	0.13	0.22	0.20
5	TVR-DTC-MD (p=0.01)	10.25	12.99	85.42	40.50	0.14	0.13	0.21	0.20

8 CONCLUSION

In this paper, we have evaluated the performance of a number of online stacking strategies, using a Java-based discrete-event simulation model. Our experiments with this model can be divided into three groups: Basic Online Rules for a Large Stack, Category Stacking, and Crane Workload. In Basic Online Rules for a Large Stack, we evaluated rules proposed in (Borgman, van Asperen, and Dekker 2010), but with a larger setup. We found that the previous conclusions all still held. In Category Stacking, we investigated an idea proposed in (Dekker, Voogd, and van Asperen 2006), wherein certain containers are interchangeable within the loading plan of a big ship. This idea proved to be working, leading to lower exit times across the board. The Crane Workload experiments showed that using the information about ASCs' task queues can be beneficial.

Our simulation model proved to be highly extensible and well-suited for the task. It will be used for future research. Topics for this could include new algorithms, for example the further investigation of Category Stacking variants, or different stack layouts, such as lanes that are parallel rather than perpendicular to the quay or with multiple ASC's per lane. Another interesting direction for future research would be to differentiate the uncertainty regarding residence times for the various modes of onwards transport as the uncertainty regarding sea-land transports is typically higher than the uncertainty regarding sea-sea transports. Finally, a comparison of these greedy, online stacking rules with optimization approaches that do look ahead, such as strategies that process the containers of an entire ship at a time, would be interesting.

REFERENCES

Borgman, B. 2009. Improving container stacking efficiency using simulation. Master's thesis, Erasmus University Rotterdam, Erasmus School of Economics. Economics & Informatics programme.

Borgman, B., E. van Asperen, and R. Dekker. 2010. Online rules for container stacking. *OR Spectrum* 32 (3): 687–716.

Dekker, R., P. Voogd, and E. van Asperen. 2006. Advanced methods for container stacking. *OR Spectrum* 28:563–586.

Duinkerken, M. B., J. J. Evers, and J. A. Ottjes. 2001. A simulation model for integrating quay transport and stacking policies in automated terminals. In *Proceedings of the 15th European Simulation Multiconference (ESM2001)*. Prague: SCS.

L'Ecuyer, P., and E. Buist. 2005. Simulation in Java with SSJ. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 611–620: Institute of Electrical and Electronics Engineers, Inc. Piscataway, New Jerssey.

Table 2: Results of experiments with Crane Workload for lane selection. The "90%" values are the average 90% percentile values. "p" is used to indicate the values for the reshuffle penalty.

Exp	Description	ROC	RDC	GPU	ASC	ETQ	ETL	90%	90%
		%	%	%	%	hrs.	hrs.	ETQ	ETL
0	RS	63.88	95.27	84.75	51.89	0.26	0.23	0.49	0.41
0	LEV	56.60	60.64	99.41	48.10	0.23	0.20	0.42	0.34
0	LEV-CW	60.43	66.03	99.37	51.77	0.19	0.18	0.31	0.28
2	LDT-DTC (unmixed, real)	6.98	7.91	88.16	41.35	0.16	0.15	0.25	0.22
2	LDT-DTC-CW (unmixed, real)	9.33	10.79	88.52	45.24	0.14	0.13	0.21	0.20
3	TVR (p=0)	52.54	81.11	92.54	45.41	0.20	0.17	0.34	0.27
3	TVR-CW (p=0)	54.04	84.17	92.38	49.23	0.18	0.16	0.27	0.24
5	TVR-DTC-MD (p=0.01)	10.25	12.99	85.42	40.50	0.14	0.13	0.21	0.20
5	TVR-DTC-MD-CW (p=0.01)	10.22	13.08	86.13	43.90	0.13	0.13	0.20	0.19
5	TVR-DTC-MD (p=0.03)	7.86	8.79	82.90	40.28	0.14	0.13	0.21	0.20
5	TVR-DTC-MD-CW (p=0.03)	7.72	8.68	83.33	43.80	0.13	0.13	0.19	0.19
6	CATS	25.27	35.33	66.89	45.46	0.18	0.19	0.30	0.30
6	CATS-CW	25.34	35.42	66.91	45.65	0.18	0.18	0.29	0.29

Park, B. J., H. R. Choi, H. K. Kwon, and M. H. Kang. 2006. *AI 2006: Advances in artificial intelligence*, Volume 4304 of *Lecture Notes in Computer Science*, Chapter Simulation Analysis on Effective Operation of Handling Equipments in Automated Container Terminal, 1231–1238. Berlin/Heidelberg: Springer Verlag.

Petering, M. E. 2010. Development and simulation analysis of real-time, dual-load yard truck control systems for seaport container transshipment terminals. *OR Spectrum* 32:633–661.

Sgouridis, S., and D. Angelides. 2002. Simulation-based analysis of handling inbound containers in a terminal. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C.-H. Chen, J. Snowdon, and J. Charnes, 1716–1724: Institute of Electrical and Electronics Engineers, Inc. Piscataway, New Jersey.

Stahlbock, R., and S. Voss. 2008. Operations research at container terminals: a literature update. *OR Spectrum* 30:1–52.

Steenken, D., S. Voss, and R. Stahlbock. 2004. Container terminal operation and operations research a classification and literature review. *OR Spectrum* 26:3–49.

AUTHOR BIOGRAPHIES

EELCO VAN ASPEREN is a lecturer at the Center for Maritime Economics and Logistics, Erasmus University Rotterdam, The Netherlands. He graduated in Business Computer Science at Erasmus University Rotterdam in 1993. He worked in IT and then as an assistant professor at the Department of Computer Science and the Econometric Institute at Erasmus University Rotterdam. In 2009 he obtained his Ph.D. in Economics at the Erasmus University Rotterdam. His research focuses on simulation with applications in logistics. You can reach him by e-mail at <vanasperen@ese.eur.nl>, and his web address is <http://people.few.eur.nl/vanasperen/>.

BRAM BORGMAN obtained his M.Sc. degree in Economics & Informatics from the Erasmus University Rotterdam and is currently a lecturer there. His email address is <borgman.bram@gmail.com>.

ROMMERT DEKKER is a full professor in operations research at the Econometric Institute of Erasmus University Rotterdam. He obtained his Ph.D. in operations research at the State University of Leiden, and his M.Sc. degree in industrial engineering from Twente University of Technology. He worked with Shell for seven years on reliability and refinery logistics. His current research interests are: maintenance and logistics (inventory control, spare parts, ports, containers and reverse logistics). He has applied simulation models in various logistical problems. His e-mail address is <rdekker@ese.eur.nl>; his web address is <http://people.few.eur.nl/rdekker/>.